

MODDARD
GRANT
IN-82-CR
219567
11P.

Final Report to
National Aeronautics and Space Administration
on
**Database Interfaces on NASA's Heterogeneous
Distributed Database System**

Shou-Hsuan Stephen Huang
Department of Computer Science
University of Houston
Houston, Texas 77204-3475

Grant NAG 5-739

July 1989

(NASA-CR-185420) DATABASE INTERFACES ON
NASA'S HETEROGENEOUS DISTRIBUTED DATABASE
SYSTEM Final Report (Houston Univ.) 11 p

CSCL 05B

N90-14133

Unclas
G3/82 0219567

[1] Summary

This is the final report on "Database Interfaces for NASA's Heterogeneous Distributed Database Systems". This document describes the syntax and semantics of all commands used in the template. Template builder should consult this document for proper commands in the template. Previous documents (Semiannual reports) described other aspects of this project. Appendix 1 contains all substituting commands used in the system. Appendix 2 includes all repeating commands. Appendix 3 is a collection of DEFINE templates from eight different DBMS's.

[2] Sample Data

We shall use the following sample clusters for all the examples in this document.

CLUSTER test1

```
TABLE s
  FIELD s#      char (5)
  FIELD sname   char (20)
  FIELD status  int
  FIELD city    char (20)
TABLE p
  FIELD p#      char (5)
  FIELD pname   char (20)
  FIELD color   char (8)
  FIELD weight  int
  FIELD city    char (10)
TABLE sp
  FIELD s#      char (5)
  FIELD p#      char (5)
  FIELD qty     int
```

CLUSTER test2

```
TABLE student
  FIELD name    char (20)
  FIELD ss#     char (9)
  FIELD gpa     real (1,2)
TABLE course
  FIELD c#      char (4)
  FIELD section char (5)
  FIELD time    char (10)
  FIELD dep     char (4)
TABLE enrollment
  FIELD c#      char (4)
  FIELD reg     int
  FIELD section char (5)
```

[2] Definitions

Instance Number: Since many data (for example, fields of a table) have multiple instance, an instance number is assigned to each one of them so as to uniquely identify these objects. Instance number always starts with integer 1. So, CLUSTER test1 will have three tables TABLE-1, TABLE-2 and TABLE-3, and 12 fields FIELD-1 (s#), ..., FIELD-12 (qty).

White Space: Newline (carriage return, end-of-line), blank, or TAB. (These are all non-printable characters.)

Command Terminator: The way the commands are scanned requires that there is a trailing whitespace (newline, balnk, or tab). This whitespace character is consumed during the scanning process and is not echoed to the filled template.

[3] Substituting Command:

3.1 Purpose: to substitute a command name with information from the primitive or system dictionary. This is the most commonly used command and probably the easiest to understand. We use this command to "fill in" information from the database.

3.2 Syntax: Any substituting command should begin with an "@" sign and followed by a string of letters. No blanks (or any other white space character) are allowed anywhere in the command. Formally,

@<name>[-<Ins.No.>]

where <name> is a string of UPPERCASE letters of size one or more and the optional <Ins.No.> is a positive integer number. Note that [xxx] is a notation for an optional component of the syntax. The reason to introduce the instance number is for the multiple occurrences of certain type of information such as fields of a table. A substituting command with an explicit sequence number is interpreted as is, i. e., no sequence numbers based on repeating commands are used (see next section for repeating command). If a substituting command without an explicit sequence number is used outside all repeating commands, the sequence number is assumed to be one.

3.3 Action: use the <name> and <Ins.No.> to find the right information to be inserted at the place where the command appears.

3.4 Example: When @RUID appears in the template (it is really @RUID-1), it should be replaced by the Resident User ID (obtained from STRUCT next_gsql).

3.5 Remark: A separate document contains a list of all legal names used so far.

[4] Repeating Command:

4.1 Purpose: to repeat a string of characters several times according to the parameter name specified. In doing so, all substituting command within this repeating command are assigned an instance number if they do not have explicit instance numbers associated with them. This is also a basic structure for building templates.

2.2 Syntax: A repeating command has a BEGIN-part and an END-part. They all begin by the "@" sign.

```
@BEGIN<name><delimiters>
    (* Arbitrary string here, including another
       nested BEGIN-END command *)
@END<name>
```

where <name> is a string of (UPPERCASE) letters and <delimiters> is defined as:

"<" <separator> [">" <L-delimiter> ">" <R-delimiter>] ">"]

The default delimiters and separator are empty strings. Notice there are several possibilities:

- o A repeating command with no delimiter at all;
- o A repeating command with only separator;
- o A repeating command with all three delimiters.

The delimiters and the separator can be any string of characters except ">". There is a way to get around this restriction (explained below). Note that blanks and other "white space" are allowed now. They will be echoed just like any other non-"<" character. In addition, to avoid messing up the template a 2-character sequence "\n" is introduced to denote a newline. As a result, the backslash character "\" is a metacharacter and to have a "\" in the string a "\\" must be used. In summary,

- (i) \n is equivalent to carriage return (newline);
- (ii) \\ is equivalent to \;
- (iii) \c is equivalent to c for any c not equal to n;
- (iv) \> can be used in the string. (So, this is how one use > in the separator or delimiter string.)

Notice this new syntax is more flexible than the old one. In effect, it allows a newline character to be used as a delimiter or separator. In fact, one can achieve the same effect in several ways. For example, to include a newline as a separator, one can do either of the followings: (A %-sign is used to indicate the newline which cannot be seen.)

```
(1)
    @BEGINTABLE%    (* % is non printable newline *)
        @TABLE %    <-- a blank before newline
    @ENDTABLE%
```

```
(2)
    @BEGINTABLE<%
    >%
        @TABLE%
    @ENDTABLE%
```

```
(3)
    @BEGINTABLE<\n>%
        @TABLE%
    @ENDTABLE%
```

In the first case, the newline separator is due to the newline following the @TABLE and the blank. Since there is a blank that serve as the terminator for @TABLE, newline is echoed. Without such a blank, newline will not be echoed.

In the second example, the newline between < and > (on two lines) are used as separator. Notice that the newline after @TABLE serve as terminator for the command @TABLE and will not be echoed.

In the last example, the newline is due to \n. This is the preferred way of using a newline as separator.

4.3 Action: Repeat the string between @BEGIN and @END according to the name of the command. For substituting commands (without explicit sequence number) inside the BEGIN-END, append proper sequence number at the end of that command. For indexing commands, convert them to proper index. (See the section on indexing commands.)

4.4 Example:

```
@BEGINFIELD<,><(><);>%
        @FIELD %    <-- blank before newline
@ENDFIELD%
```

For table S in our example, the expected result of this command is:

```
(
    s#,
    sname,
    status,
    city
);
```

Notice that "," is a separator, not a terminator. Also, when the number of columns is equal to zero, nothing should be generated, not even "(" and ");". This feature will be

used in generating the length of attribute types. Other examples:

```
@BEGINFIELD          (* no delimiters *)
    xxxxx
@ENDFIELD

@BEGINFIELD<;\n>      (* separator only *)
    xxxxx
@ENDFIELD

@BEGINFIELD<><(><)>    (* no separator *)
    xxxxx
@ENDFIELD
```

A word of caution: we do take end-of-line seriously. If we modify the first example above to:

```
@BEGINFIELD<, ><(><);> @FIELD @ENDFIELD
```

the output looks like:

```
( s#, sname, status, city );
```

In other words, if an end-of-line appears in the middle of a BEGIN-END loop and it is not consumed as the terminator of a command, it will be repeated accordingly. Also, notice that there are two blanks before @FIELD.

If a repeating command is asked to repeat a string zero times, it simply ignores the command. Thus, no delimiters or separators will be generated in the output.

4.5 Remark: A separate document contains a list of legal names used so far.

[5] Indexing Command:

5.1 Purpose: to provide a way to access the indexes of a (nested) repeating command. This is used less often than the other two.

5.2 Syntax: @<level> where <level> is a single digit positive integer number.

5.3 Action: Replace the index of the repeating command based on the parameter of the command. The parameter <level> indicates the level of nesting of the repeating command with the outermost level equal to one.

5.4 Example:

```
@BEGINCLUSTER
```

Appendix 2: Repeating Command Names

Name	Meaning
-----	-----
CLUSTER	No. of result cluster
TABLE	No. of result tables in a cluster
FIELD	No. of result fields in a table
LENGTH	No. of length indicators in a result field
SCLUSTER	No. of source cluster
STABLE	No. of source tables in a cluster
SFIELD	No. of source fields in a table
SLENGTH	No. of length indicators in a source field
WHERE	No. of where clauses

Appendix 3: DEFINE Templates of Eight Different DBMS's

[1] Ingres

Template:

```
====
@BEGINTABLE
create @TABLE (
@BEGINFIELD<,\n>
@FIELD = @TYPE @BEGINLENGTH @LENGTH @ENDLENGTH
@ENDFIELD )
@ENDTABLE
=====
```

The Filled Template:

```
====
create employee(
emplname = c20,
emplid = i2)
create deptment(
emplid = i2,
deptid = c10)
=====
```

[2] DB2

Template:

```
====
create database @DBNAME
@BEGINTABLE
create table @TABLE
(@BEGINFIELD<,\n>
@FIELD @TYPE @BEGINLENGTH<><(><)> @LENGTH @ENDLENGTH
@ENDFIELD )
in database @DBNAME-1
@ENDTABLE
=====
```

The Filled Template:

```
====
create database mydbase
create table employee
(emplname char(20),
emplid int)
in database mydbase
create table deptment
(emplid int,
deptid char(10))
in database mydbase
=====
```


[3] System R

Template:

```
====  
@BEGINTABLE  
create table @TABLE  
(  
  @BEGINFIELD<,\n>  
  @FIELD (@TYPE @BEGINLENGTH<><(><)> @LENGTH @ENDLENGTH )@ENDFIELD )  
@ENDTABLE  
=====
```

The Filled Template:

```
====  
create table employee(  
  emplname (char(20)),  
  emplid (integer))  
create table deptment(  
  emplid (integer),  
  deptid (char(10)))  
=====
```

[4] Unify

Template:

```
====  
@BEGINTABLE  
@TABLE  
@BEGINFIELD  
@FIELD @TYPE @BEGINLENGTH @LENGTH @ENDLENGTH  
@ENDFIELD  
@ENDTABLE  
=====
```

The Filled Template:

```
====  
employee  
emplname string 20  
emplid numeric 10  
deptment  
emplid numeric 10  
deptid string 10  
=====
```

[5] Knowledgeman

Template:

```
====
@BEGINTABLE
define @TABLE
@BEGINFIELD
@FIELD @TYPE @BEGINLENGTH @LENGTH @ENDLENGTH
@ENDFIELD
@ENDTABLE
=====
```

The Filled Template:

```
====
define employee
emplname str 20
emplid num
define deptment
emplid num
deptid str 10
=====
```

[6] Sabrina

```
====
@BEGINTABLE
create @DBNAME-1 .@TABLE (
@BEGINFIELD<,\n>
@FIELD : @TYPE @ENDFIELD )
@ENDTABLE
=====
```

The Filled Template:

```
====
create mydbase.employee(
emplname: text,
emplid: integer)
create mydbase.deptment(
emplid: integer,
deptid: text)
=====
```

[7] Informix

Template:

```
====
@BEGINTABLE
database @DBNAME-1
file @TABLE
@BEGINFIELD
field @FIELD type @TYPE @BEGINLENGTH<>< length ><> @LENGTH @ENDLENGTH
@ENDFIELD
end
@ENDTABLE
=====
```

The Filled Template:

```
====
database mydbase
file employee
field emplname type character length 20
field emplid type integer
end
database mydbase
file deptment
field emplid type integer
field deptid type character length 10
end
=====
```

[8] Focus

Template:

```
====
filename=@DBNAME , suffix = foc,$
@BEGINTABLE
segname=@TABLE , segtype= s1,$
@BEGINFIELD
fieldname=@FIELD , format=@TYPE @BEGINLENGTH @LENGTH @ENDLENGTH ,$
@ENDFIELD
@ENDTABLE
=====
```

The Filled Template:

```
====
filename=mydbase, suffix = foc,$
segname=employee, segtype= s1,$
fieldname=emplname, format=a20,$
fieldname=emplid, format=i2,$
segname=deptment, segtype= s1,$
fieldname=emplid, format=i2,$
fieldname=deptid, format=a10,$
=====
```